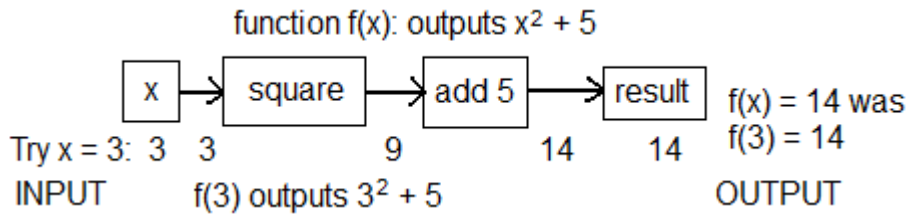
Math for Programmers: Functions

I am a programmer; welcome to my world! Today, you will get a taste of what I do for a living and in my spare time. The original handout that I was going to give you was starting to drag on for over six pages, and was using too much complicated computer jargon, so I'm going to give you all a simplified version. You're not here to learn the first few weeks of programming, at a highly technical level, in two hours; you're here to ace your SAT, so that you can get into college, so that maybe THEN, you'll major in computer science ….

A computer is just a really fast calculator, with lots of space to store information (in memory and on disks). Conversely, a calculator is just a really slow computer, with only a little bit of space to store information.

One of the things that we programmers do is we write functions that the computer can understand, and use, to calculate things for us. This is a diagram of a function:



This is that same function written as something resembling the *source code* of a program, in some programming language:

```
function f(x)
{
    output ((x) * (x)) + 5;
}
```

Note: On the computer, `*` is often used for multiplication, and `/` is often used for division.

The function's *domain* is the set of all possible values of the input *parameter*, "x". These are all of the values that "make sense" as input. Here, the domain consists of the set of all real numbers. The function's *range* is the set of all possible values that it can output. Here, the range is all real numbers greater than or equal to 5. A function *maps* its domain to its range. Those parentheses are not necessary, by the way, given order of operations, but I'll use them later for something fun.

Did you know that all mathematical *operators* can be expressed as functions, though? Imagine division:

```
function ÷(a, b)
{
    output (a) / (b);
}
```

The *expression*, "a ÷ b", becomes ÷(a, b). This works for ANY funky operator-like symbol the SAT might throw at you, such as Щ, ☺, or ♫. The expression, "4 ♫ x", becomes ♫(4, x).

Function "÷" has an *arity* of two, meaning that it takes two parameters. A function with an arity of two is often called a *binary* ("2-ary") function. Function "f" had an arity of one, and can be called a *unary* ("1-ary") function. Can anyone tell me the domain and range of this function? Each parameter has its own domain. What is the arity of the following function:

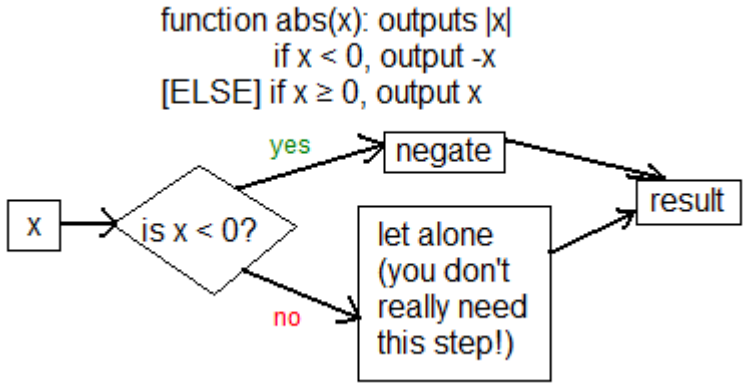
```
function one()
{
    output 1;
}
```

Yes, you can have a *nullary* ("0-ary") function, or one with an arity of zero. It doesn't really have a domain, but its range is the set {1}.

Now, let's try something fun:

```
function abs(x)
{
    if x < 0: output -(x);
    else: output (x);
}
```

How many of you recognize this function? It's the absolute value function. What is its domain and range? I'll show you a diagram of how it works:



Try navigating the flowchart, with various values of "x".

Now, on to the last fun thing we are going to do today: *composition*! You can input anything to a function that is within that function's domain. You can also input a function to another function, in order to *compose* the two functions. Let's say we have $f(x) = x^2 + 5$ again. We also have $g(x) = x + 1$. Now, we will calculate f(g(x)) ("f" of "g" of "x"). Sometimes, this is written as $(f \circ g)(x)$. Those parentheses will come in handy now:

```
function f(x)
{
    output ((x) * (x)) + 5;
}
```

Look at f(x + 1):

```
((x + 1) * (x + 1)) + 5;
```

After using the FOIL method, f(g(x)), or f(x + 1):

$x^2 + 2x + 6;$

It can also be seen as:

```
(x * x) + (2 * x) + 6;
```

Substitute the VALUE OF the "x" parameter for the "x". For example, if "x" = 4, then replace every "x" with a 4. Try finding f(g(10))? What about g(f(x)), and by extension, g(f(10))?

* I hope this put the FUN in functions! :-)